**Chapter 3 – Lists**

Lists can be used in many ways and covering the various methods of using them would take too long. What we will do is explore key features that can be used for our text based game.

Lists are a type of data structure which is often used in programming. They are very similar to what the name suggests – a list of variables. The size of this list changes depending on how many variables are stored within it. Let’s say for example, we want to find the 5th element in our list, we can use indexing to get the data that is stored within it. It would look like this:

5thelement = List[4]

There are quite a few ways of adding elements to our list. This first method is to explicitly declare each element and their order. Another method is to use the keywords “append” or “extend” or “insert”.

Explicit declaration

myList = [1,2,3,4,5]

If we call our list it will simply return [1,2,3,4,5].

Using append

myList.append(6)

We use append when we want to add exactly one element to our list. When we call our list now it will say [1,2,3,4,5,6].

Using extend

myList.extend([7,8,9])

If we choose to add multiple elements then extend is the keyword to use. Our list now becomes [1,2,3,4,5,6,7,8,9].

Using insert

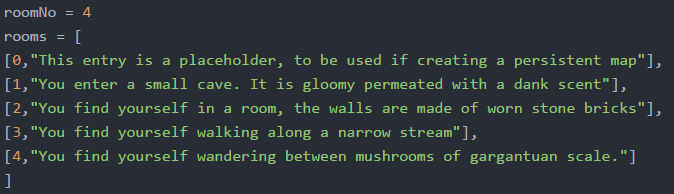
myList.insert(1,1.5)

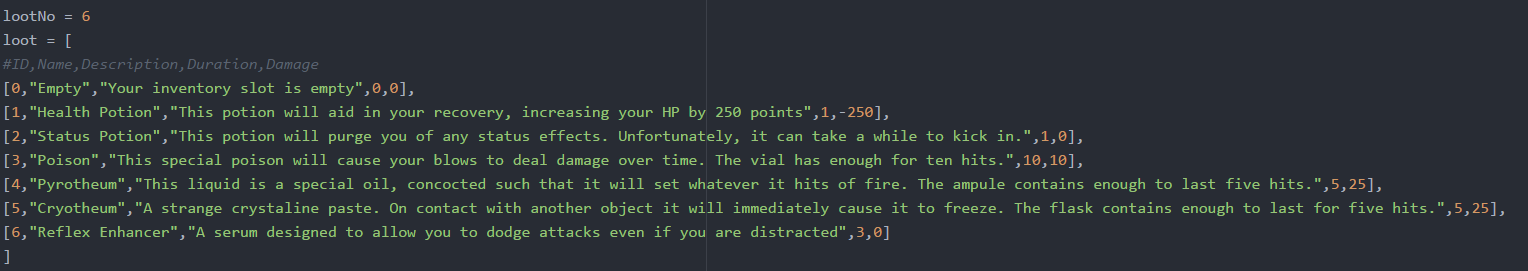
With this the 2nd value in the brackets is the item to be added and the first number is the positon to which the element will be added. It is important to note that the indexing always starts on 0, so therefore our new list is [1,1.5,2,3,4,5,6,7,8,9]

There are various ways to find out information from our lists which we can use in conjunction with if statements which we learnt about in **chapter 2.**

Example 3.1

Our game will include multiple rooms and we want to have a different description for each room. In this example we will go one step first by putting lists within lists. For now, we have four types of rooms and each room will have a room number and a description. The room number is helpful for the programming side of things so that we can have a way of referencing each room. The description will be a string of characters which will be using every time a player enters a new room. Here is the example from our code:



Here is a more complex list but once we explore each element of the list, it becomes very simple:

Notice we have a comment in this extract. Comments are very useful as they add small labels to help you as the programmer to understand what the code does at a quick glance. Comments have to start with #.

Looking at the content of this list, it contains six smaller lists each of length 5. If we look at the comment we put, each nested list follows the property, The first element is the ID of the loot item, the second is the name of the item, third is a description of the item and the last 2 elements represent duration and damage specific to the item.

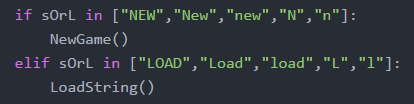
This list will be used to generate loot and at this current point in time we have 5 items as well as 1 slot to represent the chance of getting no loot.

We will be using lists further on in this tutorial for things such as storing our monster database.

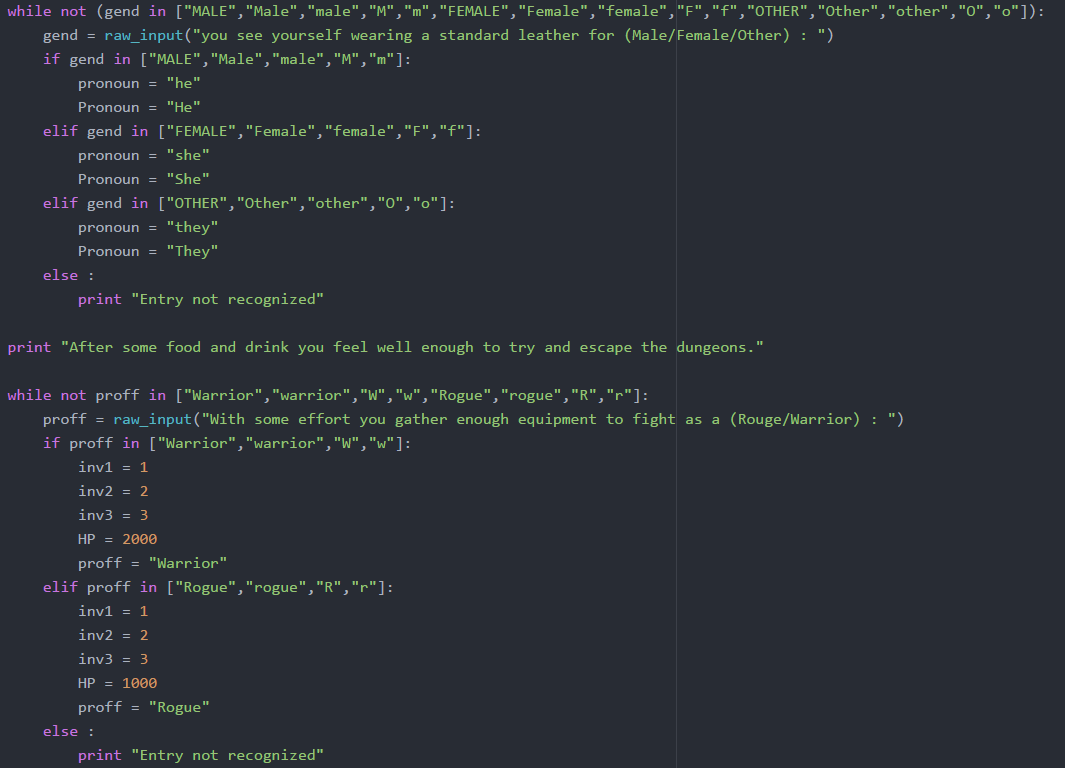
Example 3.2

Lists will be used a lot with if statements in our game this will be mainly for recognised player input. We want to make sure that our game recognises most things that a player is likely to input. However, in all programming languages the string “Hello” is not the same as “hello” simply due to one having an upper-case h. Therefore, we want to make sure that our game recognised variations of the same input. We may not cover every single input but we should make sure to cover the major ones.

We can do this by following the format *if variable in list.* What this does is checks if a variable is an element of a list and executes like a normal if statement whether it is true or false. This way we can crate lists with variations of words and here are a few examples:

’

If we combine everything we have learnt from all three chapters you should be able to interpret the following which is an excerpt from our character creation. Try to have a go at understanding it now and we will then explain in more depth once we reach the chapter dedicated to starting a new game and character.



MINI EXERCISE

1. Write a small program that asks for 10 numbers and puts them into a list. After the list is complete you must print the highest number in the list.
2. Write a small program that takes in a list of 10 numbers. Produce two new lists for odd and even numbers and print them. (hint: look at a modulo operator)
3. Returning to the shopping list problem from the previous chapter, store all the items in a list and then print the full list once the user has finished.